design ideas

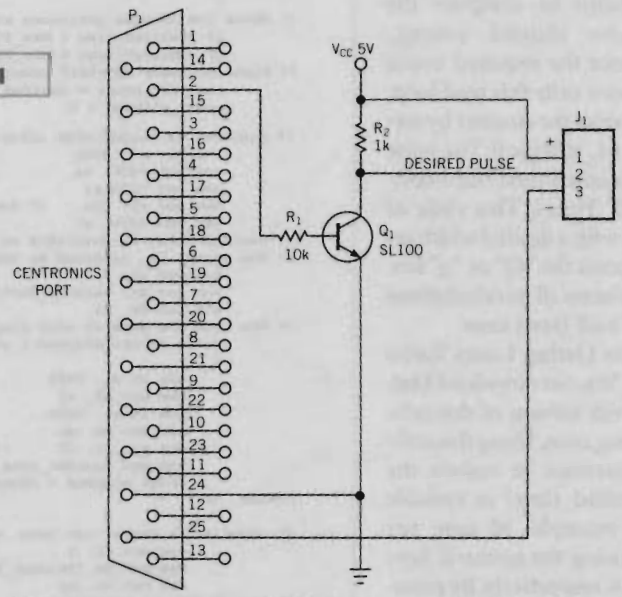
Edited by Bill Travis and Anne Watson Swager

Centronics port generates narrow pulse widths

Darvinder Oberoi, CEDTI, Jammu, India

VARIABLE-PULSE-WIDTH signals are useful in control circuitry for positioning and holding purposes in robotics and power electronics. Frequently, the need arises for pulses with width less than 1 msec. Delays less than 1 msec are usually not available in most programming languages, so generating such pulses can be a problem. To generate a fractional-millisecond delay you can use a PC's 8254 16-bit timer (Counter 2), which normally controls the PC's speaker. The desired pulse is available at the PC's Centronics port (**Figure 1**) through a buffer stage, which protects the port from overload damage. Counter 2 operates at a clock frequency of 1.193181 MHz. To generate a pulse width less than 1 msec, you operate Counter 2 in mode 2 as a rate generator. You do this by setting the control-word value to $0B4_h$ and by writing this data to the control-register port, address 43_h. Initially, Counter 2 contains $FFFF_h$ at address 042_h. Bit 0 of port 61_h is at logic 1 to enable the counter, and bit 0 of the printer port (in this case, 0378_h, printer_port) is at logic 0. **Listing 1** contains the software necessary for controlling the pulse-generation process.

Setting bit 0 of port 61_h enables Counter 2. The counter decrements by



You can use the Centronics port on your PC to generate narrow pulse widths.

LISTING 1—TURBO C11 LISTING FOR PULSE GENERATION

(Continued on next page)

```
#include <dos.h>
#include <conio.h>
#include <stdlib.h>
#pragma inline

/*** All values are in milli Second  ***/
#define Max_Time      0.750   /* Maximum Pulse Width Time(mS)        */
#define Neutral_Time 0.500    /* Neutral Width Time(mS)              */
#define Min_Time      0.250   /* Minimum Pulse Width Time(mS)        */
#define Time_Step     0.050    /* Time(mS) Steps for Pulse Change      */
#define Counter_Count_Time 0.0008380958 /* Counter2's single count period(mS) */
#define Off_Time  10              /* Pulse OFF Duration (mS)
*/

/* Address of Printer Port     */
#define Printer_Port 0x378
void screen_display();         /* Routine for displaying Instruction */
/***** Main Program Starts Here *****/
main()
{
      char ch;
      unsigned counter_data, count_elapsed,required_count;
      float desired_time,counter_time;
      desired_time=Neutral_Time;   /* Initial time                */
      clrscr();
      screen_display();
      /* Counter2 is set in Mode 2, control word  data is b4        */
      asm mov al, 0b4h       /* Set the counter2 operation */
      asm out 043h,al
      for (;ch != 'q';)    /* Loop Till q/Q key is pressed */
      {
       if (kbhit())        /* Check, if any key has been pressed or not */
         ch = getch();
         switch (ch)         /* And take action depending upon the key pressed */
         {
```

one every 0.8380958 µsec. Before the generation of the pulse, it's necessary to compute Counter 2's required count (required_count) to generate a pulse of desired duration (desired_time). When the counter is enabled, a "while" loop reads back the counter's count through port 42_h in two read cycles. This count (counter_data) data helps to compute the counts that have elapsed (count_elapsed), and, once the required count arrives, the software exits this read loop. The software disables the counter by setting bit 0 of port 61_h to logic 0. The pulse goes low for the desired time (for example, 10 msec Off_Time). This cycle of pulse generation with a desired width repeats until you press the "Q" or "q" key. The software performs all its calculations during the pulse's off (low) time.

The software in **Listing 1** uses Turbo C++, Version 3. You can download **Listing 1** from the Web version of this article at www.ednmag.com. Using this software, you can increase or reduce the pulse width (desired_time) in variable time steps (for example, 50 µsec per Time_Step) by using the numeric keypad's keys 6 and 4, respectively. By pressing Key 5, you can fix the pulse's duration at a nominal value (for example, 500 µsec Neutral_Time). Key 8 fixes the pulse's duration at the maximum desired pulse width (for example, 750 µsec Max_Time). Key 2 fixes the pulse's minimum desired width (for example, 250 µsec Min_Time). The hardware we used for testing the software is a P-II system running at 400 MHz, with 32 Mbytes of RAM, operating in MS-DOS mode.

Is this the best Design Idea in this issue? Vote at www.ednmag.com/ednmag/vote.asp.

LISTING 1—TURBO C11 LISTING FOR PULSE GENERATION (Continued)

```
                case 'Q': ch = 'q';break; /* If Q key is     pressed */
                case 52: desired_time -=Time_Step; /* If 6 NUM Key is pressed */
                   ch=' ';break;
                case 54: desired_time += Time_Step;/* If 4 NUM Key is pressed */
                   ch=' ';break;
                case 53: desired_time=Neutral_Time; /* If 5 NUM Key is pressed */
                   ch=' ';break;
                case 56: desired_time =Max_Time; /* If 8 NUM Key is pressed */
                   ch=' ';break;
                case 50: desired_time =Min_Time; /* If 2 NUM Key is pressed */
                   ch=' ';break;
        }
/* Check the extreme positions and reset them in case of violation  */
        if (desired_time > Max_Time)  desired_time = Max_Time;
        if (desired_time < Min_Time)  desired_time = Min_Time;
/* Finds how many counter2 counts are required for the  desired time interval */
        required_count = desired_time/Counter_Count_Time;
        count_elapsed = 0;

/* Counter2 is loaded with initial with  count of ffff            */
         asm mov al, 0ffh          /* Load the initial count as ffff */
         asm out 042h, al          /* Done in two cycles          */
         asm out 042h,al
         asm mov al, 01h    /* Enable the counter to Start the ON period */
         asm out 061h, al
/* Desired Pulse is available at printer port's pin no.-2 */
/* The signal is  inverted by the buffer stage            */
         asm mov al, 0
         asm mov dx, Printer_Port
         asm out dx, al
/* The read the data in this loop, till desired delay is obtained        */
         while (count_elapsed < required_count)
         {
             asm in al, 042h       /*  Get LSB        */
             asm mov cl, al
             asm in al, 042h           /*  Get the MSB    */
             asm mov ah, al
             asm mov al, cl     /* Two bytes, combined to get 16 bit data */
             asm mov counter_data , ax
             count_elapsed = (0xffff - counter_data);  /* Calculate the elapsed
counts  */
         }
/*  Make pulse signal low here, after the desired counts have elapsed  */
             asm mov al, 1             /* Signal is inverted */
             asm mov dx, Printer_Port
             asm out dx, al
/* Disable the counter2 here */
             asm mov al, 00h
             asm out 061h, al
/* Print the information on the screen */
             counter_time=count_elapsed*Counter_Count_Time; /* Calculate counter
time */
             gotoxy(8,12); cprintf("%f",desired_time);
             gotoxy(26,12); cprintf("%f",counter_time);
             gotoxy(45,12); cprintf("      ");
             gotoxy(60,12); cprintf("      " );
             gotoxy(45,12); cprintf("%u", count_elapsed);
             gotoxy(60,12); cprintf("%u",required_count );
/* OFF period taken as 20ms*/
             delay (Off_Time);
      }                    /* End of FOR Loop */
      textcolor(LIGHTGRAY);   /* Set the text color and clear the screen */
      clrscr();
}                          /* END of the MAIN program here */

/**** Routine Which display the instructions & Other Information   ****/
void screen_display()
{
      textcolor(YELLOW);
      gotoxy(26,2); cprintf("Make Sure Num Lock is ON");
      gotoxy(5,11); cprintf("Desired Time(mS)  Counter Time(mS)  Count Elapsed
Required Count");
      textcolor(YELLOW);
      gotoxy(25,19); cprintf("< Press Q/q key to Quit the Program > ");
      gotoxy(15,21); cprintf("Press 4 NUM Key to Decrease Time        (By %f
ms)",Time_Step);

      gotoxy(15,22); cprintf("Press 6 NUM Key to Increase Time        (By %f
ms)",Time_Step);
      gotoxy(15,23); cprintf("Press 8 NUM Key For Maximum Pulse Width ( %f ms
)",Max_Time);
      gotoxy(15,24); cprintf("Press 2 NUM Key For Minimum Pulse Width ( %f ms
)",Min_Time);
      gotoxy(15,25); cprintf("Press 5 NUM Key For Neutral  Position   ( %f ms
)",Neutral_Time);
      textcolor(WHITE);
}
/*pulse.c*/
```

Preprocessor for rotary encoder uses PAL

David Rathgeber, Alles Corp, Toronto, ON, Canada

ROTARY ENCODERS usually provide quadrature pulses that indicate both the amount of rotation and the direction (**Figure 1**). A microcontroller can calculate the rotation direction and keep track of angular movement. Many microcontrollers' interrupt inputs, such as those on the Zilog Z86C90, can detect only a falling edge. Some with programmable edge detection, such as the Zilog Z86E30, can function with either rising or falling edges but not both. However, for maximum resolution, it is desirable to look at each rising and falling edge. In these cases, you need four inputs to read the encoder. Further, when you read the two inputs, you can assess the direction of rotation only by referring to the previous read operation. The microcontroller thus must keep track of the previous state of each input. You can use a simple PAL or other programmable-logic chip to precondition the encoder outputs to produce a single falling-edge output for each rising and falling edge for each rotation direction.

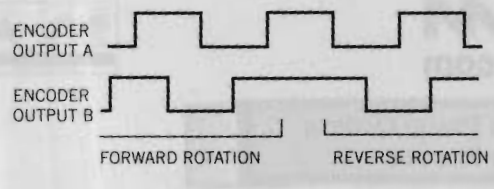
LISTING 1—PAL SOFTWARE FOR ENCODER PREPROCESSOR

```
"INPUTS
STROBE          PIN     1;
A               PIN     2;      " ENCODER
B               PIN     3;      " ENCODER


"OUTPUTS
                "DELAYED ENCODER INPUTS
ADEL    PIN 14  istype 'reg_d';
BDEL    PIN 15  istype 'reg_d';
                "FORWARD AND REVERSE INTERRUPTS
FORWARD         PIN 18  ISTYPE 'REG_D';         "FORWARD IRQ TO CPU
REVERSE         PIN 19  ISTYPE 'REG_D';         "REVERSE IRQ TO CPU


EQUATIONS
                "1. LATCH 2 INPUTS ON RISING EDGE OF STROBE
ADEL.CLK        = STROBE;
ADEL.D          = A1;
BDEL.CLK        = STROBE;
BDEL.D          = B1;

                "2. COMBINE TO FORM IRQ
!FORWARD.CLK = STROBE;

!FORWARD.D =    (A & ADEL & !B & BDEL)
                # (!A & ADEL & B &!BDEL)
                # (!A & !ADEL & B &  !BDEL)
                # (A & !ADEL & B &  BDEL)

!REVERSE.CLK = STROBE;

!REVERSE.D =    (!A & ADEL & B & BDEL)
                # (!A & !ADEL & !B & BDEL)
                # (A & !ADEL & !B & !BDEL)
                # (A & ADEL & B &  !BDEL)

END
```

This design uses an AMD PALCE-16V8Z that requires minimal supply current and costs less than $1. In addition to the two encoder inputs, the scheme requires a strobe, which occurs much more frequently than the encoder edges. This design uses the $\overline{DS}$ pin on the microcontroller, but you could probably use the crystal clock as well. Two PAL outputs are the negative-going direction outputs, and two other outputs serve as D registers. A single chip can thus condition two encoders with inputs and outputs left over for miscellaneous encoding tasks. The design works as follows: The two extra outputs serve as D-type registers, with an encoder input as the data and the strobe serving as the clock. When an input (A or B in **Figure 2**) changes, the corresponding delayed output (ADEL or BDEL) changes on the next strobe. Therefore, the delayed outputs mirror the inputs, with the delay depending on the strobe's frequency. An examination of the diagram in **Figure 2** reveals four unique patterns that relate to forward encoder rotation and four that relate to reverse rotation. **Listing 1** shows the PAL software, in Synario format. You can download the PAL software from the Web version of this article at www.ednmag.com.

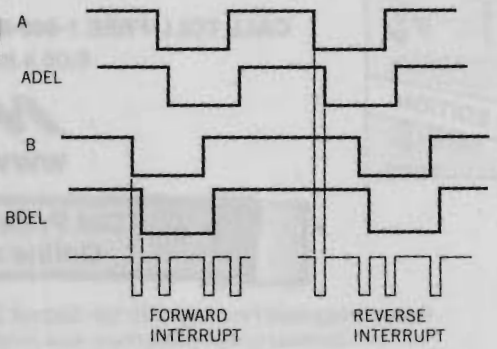
Is this the best Design Idea in this issue? Vote at www.ednmag.com/ednmag/vote.asp.

Figure 1



To determine the direction of rotation, it's necessary to detect both rising and falling edges of the encoder outputs.

Figure 2



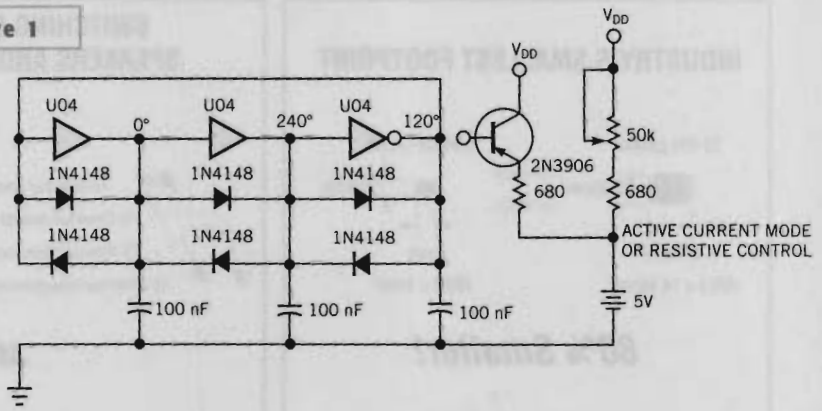
A PAL produces unique patterns for forward and reverse rotation.

Inverters form three-phase VCO

Al Dutcher, AL Labs, West Deptford, NJ

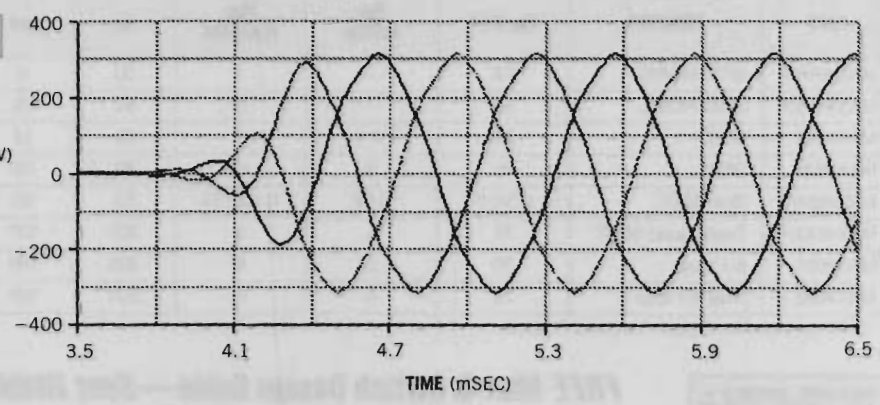
YOU SOMETIMES NEED an inexpensive VCO that can produce evenly spaced three-phase outputs over a wide frequency range. You could use tracking all-phase filters with only one oscillator, but this technique is difficult to implement and offers limited range. Other methods, such as using a DSP, are feasible, but they're complex and expensive. The inspiration for the VCO in **Figure 1** came from Texas Instruments' application notes of years ago, detailing the use of unbuffered U-type inverters for use in ring oscillators. The application note's circuit consists of only the inverters. The circuit generates relatively squarish waveforms. Any ring oscillator's operation depends on the fact that an odd number of inversions exists around the loop. Any odd number of inverters would work. The feedback is inverting, or negative. The feedback creates an initial bias equilibrium at the transition voltage for the gates.

Figure 1



A ring-type oscillator generates 3-phase outputs over a wide frequency range.

Figure 2



The circuit in Figure 1 generates 120°-spaced, 600-mV p-p outputs.

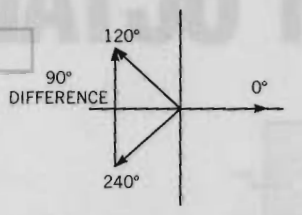
Loop gain greater than unity is a necessary condition for oscillation. Unbuffered inverters typically have a gain of 15 or thereabouts at dc and approximately 7 with capacitive loads. Three inverters thus have a total gain of more than 340, which is plenty for oscillation. At high frequencies, the inverters exhibit a lagging phase shift arising from propagation delay. Enough lagging shift added to the inversions ultimately turns the total inversion into noninversion. The circuit of **Figure 1** starts out with the 180° inversion and adds 60° of lag for 240° per stage. Three stages of 240° works out to 720° total. This figure represents two complete trips around the phase circle for noninversion. Noninversion implies regeneration, which begets oscillation. With no added capacitors, the circuit of **Figure 1** can operate at frequencies as high as tens of megahertz using 74ACU04 gates. Added capacitors can drop the frequency to usable levels. The frequency equates to $I_{DD}/3C$. For lower frequency applications, 74HCU04 gates are suitable, because they're less sensitive to layout considerations.

The diodes in **Figure 1** perform two tasks. First, they limit the excursions of the gates to 600 mV p-p, so that the gates always operate in their linear region. Second, they allow the gates to operate as current diverters, alternately charging and discharging the capacitors. The rate at which the capacitors discharge depends on the common supply current to the inverters. This rate and, hence, the oscillation frequency is proportional to the operating current. The range of frequencies over which the circuit can operate is more than 1000-to-1 (supply current from 10 μA to 10 mA). Note that at the low-current, low-frequency end of the range, the circuit cannot supply much

signal current and might need buffering. **Figure 2** shows the three-phase waveforms. The concept doesn't work well with normal ac- or HC-buffered gates, because they have far too much gain and would drive the nodes into square waves. The ACU and HCU types are somewhat obscure, but they nonetheless have multiple sources. Don't forget to ground the inputs of the remaining three gates of these hex devices. Floating inputs are verboten with all CMOS devices.

The circuit generates three equally spaced outputs. Because it generates substantially sinusoidal outputs, you can easily obtain quadrature-spaced outputs by trigonometric means (**Figure 3**). You can connect a differential amplifier to the 120 and 240° outputs. It rejects the common-mode, 180° components. The difference between these two outputs is at 90°, and it tracks well over the range of the oscillator. You can use the same type of differential amplifier to amplify the 0° output so that the amplifier delays track at higher frequencies. In principle, you can extract any set of phase angles by the judicious adjustment of component amplitudes in the external circuits. The circuit's main drawback is that it does not have a high-Q resonator attached, so phase noise could be a problem. When you incorporate the circuit into a relatively tight PLL circuit, the performance improves considerably. The capabilities of this oscillator allow it to lock over a wide range of frequencies.

Figure 3



Vector subtraction of two outputs produces a 90° quadrature signal.

Is this the best Design Idea in this issue? Vote at www.ednmag.com/ednmag/vote.asp.

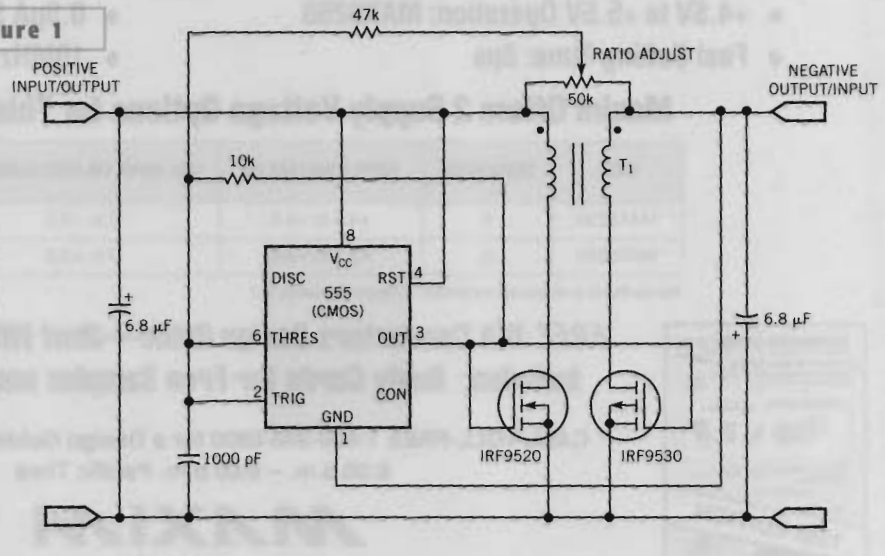
Power inverter is bidirectional

Tom Napier, North Wales, PA

If you want to swap charge in either direction between unevenly loaded positive and negative battery buses, you need an inverting dc transformer. One implementation is the symmetrical flyback converter shown in **Figure 1**. The circuit can generate a negative output from a positive supply or a positive output from a negative supply. When the circuit starts up, the substrate diode of the output FET bootstraps the output voltage to the point where synchronous switching takes over. When the gate-switching signal is symmetrical, the output voltage is approximately −95% of the input voltage, and the efficiency is greater than 80%. You can obtain voltage step-up or step-down by adjusting the switching ratio.

When I used the circuit between two 4V lead-acid batteries, a comparator adjusted the switch ratio to drive charge in the desired direction. The circuit automatically replaces charge drained from one battery to the other. In a short-battery-life application, the 2.5-mA standby current from each battery may be negligible. Using lower-gate-capacitance, FETs can reduce losses. Alternatively, you can add gates to the drive circuit to turn off both FETs whenever the battery voltages balance. The minimum input voltage is a function of the gate thresholds of the FETs. The ±9V rating of the CMOS 555 timer sets the maximum voltage. My prototype supplies approximately 100 mA.

Figure 1



An inverter circuit swaps charges between opposite-polarity batteries.

Is this the best Design Idea in this issue? Vote at www.ednmag.com/ednmag/vote.asp.

SP6121

Superb Performance, Simple Design

features

- Resistor programmable V_{OUT} (1.25V to7V)
- < 1% Initial reference accuracy
- Optimized for 3A to 10A output current with 95% efficiency possible
- Compact surface mount solution
  - No external charge pump or soft start circuitry needed
  - High frequency (500kHz) uses smaller components
- 0% to 100% duty cycle ensures low dropout during low V_{IN} condition
- Fast transient response supports high speed dynamic loads
- 500µA, quiescent current
- Single input supply operates down to 3V
- Fully fault protected
  - Lossless adjustable current limit with high side RDS(on) sensing
  - Hiccup mode current limit protection
  - Over and under voltage protection

applications

- Supply bias for high speed
  - DSP's, dynamic logic
  - Microprocessor cores
  - Digital ASICs
- Video and multimedia cards
- SIP modules for distributed power systems

Evaluation board & samples available

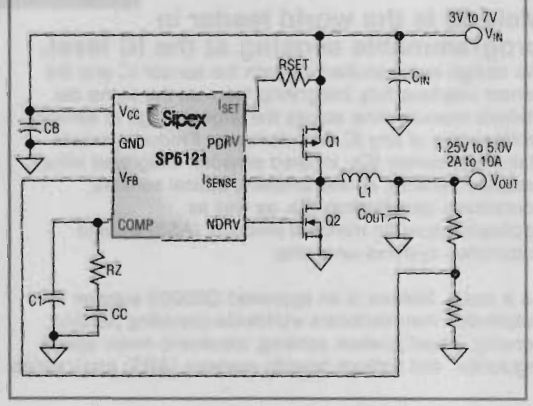
available package

8 lead nSOIC

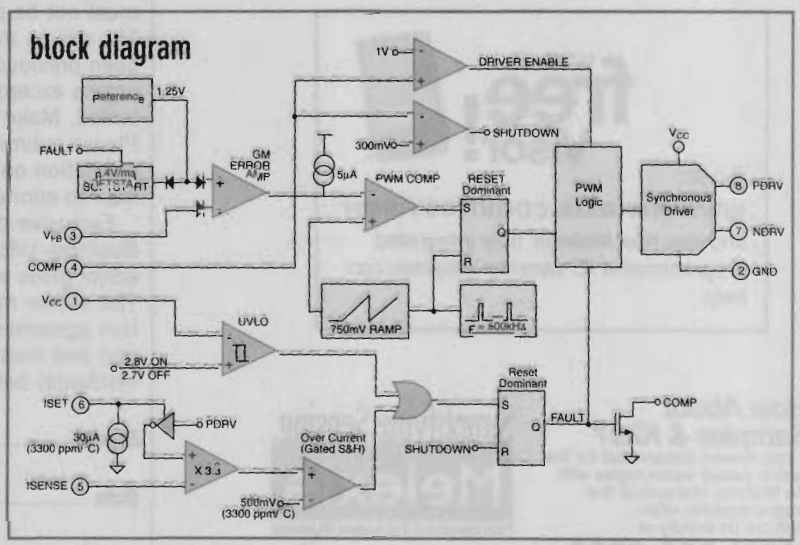
A Low Voltage, High Current, Synchronous PWM Buck Controller that's Small, Flexible, and Cost Effective

The SP6121 provides a high level of performance to low voltage (≥1.25V), high current dynamic loads without sacrificing solution size, performance, accuracy or ease of use. The 500kHz fixed switching frequency uses a small surface mount inductor and capacitors and the integrated soft start circuit and PFET switch minimize component count. The synchronous buck topology easily delivers efficiencies of 95% and the on chip reference ensures high output accuracy over line, load and temperature. By implementing a fixed frequency topology and bringing the soft start function on chip, a designer can quickly create a robust low voltage, high power solution using only a few easily chosen external components.

typical application diagram



block diagram



Available Exclusively From:

FUTURE ELECTRONICS

1-888-441-0183

Sipex

sipex.com/newproducts

Enter 52 at www.ednmag.com/info